POZNAN UNIVERSITY OF TECHNOLOGY

Wojciech Mioduszewski

Klasyfikacja danych opisanych za pomocą szeregów czasowych

Master's Thesis

Supervisor: dr inż. Jerzy Błaszczyński

Poznań, 2015

Spis treści

Rozdział 1

Wstęp

Cel i zakres pracy

Cel: Opracowanie i implementacja różnych podejść do klasyfikacji danych czasowych.
Zadania:

- Zapoznać się z literaturą tematu.
- Opracować wybrane podejścia do klasyfikacji danych czasowych.
- Zaimplementować i udokumentować zaproponowan rozwiązania.
- Przeprowadzić eksperyment obliczeniowy

Początkowo celem niniejszej pracy była analiza szeregów czasowych zawierających dane ciśnienia w oku pacjentów zdrowych, oraz tych ze zdiagnozowaną jaskrą. Ponadto zamiarem było użycie do tego celu metody SAX, a następnie zbudowanie klasyfikatora potrafiącego sklasyfikować dane wytworzone przez tą metodę. Równie ważne było to, aby nie testować sposobów klasyfikacji tylko i wyłącznie na danych zebranych w celu oceny jaskry, lecz również sprawdzić jak wybrane i stworzone metody poradzą sobie w odniesieniu do innych szeregów czasowych. Kolejną rzeczą, od której należało się uniezależnić są klasyfikatory, dlatego też eksperymenty przeprowadzone zostały na kilku różnych technikach kategoryzowania instancji.

The goal and the scope of the thesis

Celem pracy jest opracowanie / wykonanie analizy / zaprojektowanie / ..... Struktura pracy jest następująca. W rozdziale 2 przedstawiono przegląd literatury na temat ..... Rozdział 3 jest poświęcony ..... (kilka zdań). Rozdział 4 zawiera ..... (kilka zdań) ..... itd. Rozdział x stanowi podsumowanie pracy.

Rozdział 2

Background

2.1 Definicja szeregu czasowego

Szereg czasowy jest to seria pewnych obserwacji osadzonych w czasie. Można powiedzieć, że jest to przyporządkowanie danych liczbowych do odpowiadających im punktów w czasie, najczęściej z jednakowymi odstępami między kolejnymi wartościami.

http://www.cs.put.poznan.pl/jstefanowski/aed/TPtimeseries.pdf

2.2 Przykładowe metody analizy danych czasowych[6]

Tak obszerny problem jak analiza danych czasowych musi nieść za sobą rozmaite metody przeprowadzania tej analizy. Pomimo tego, że do ostatecznego eksperymentu wybrano tylko jedną z nich - regresję, to postanowiono przedstawić po krótce czym charakteryzują się poszczególne metody oraz ewentualnie dlaczego zostały porzucone.

2.2.1 Regresja liniowa

Regresja w odniesieniu do danych czasowych sprowadza się do estymowania liniowego trendu jaki prezentuje badany szereg. Koncepcyjnie metoda polega na stworzeniu funkcji liniowej, która w najbardziej dokładny sposób przybliża wartości na kolejnych obserwacjach. Matematyczny model regresji ma zatem następującą postać:

$$y = ax + b$$

Jako miarę błędu przyjmuje się sumę kwadratów różnicy między oszacowaniami, a wartościami właściwymi.

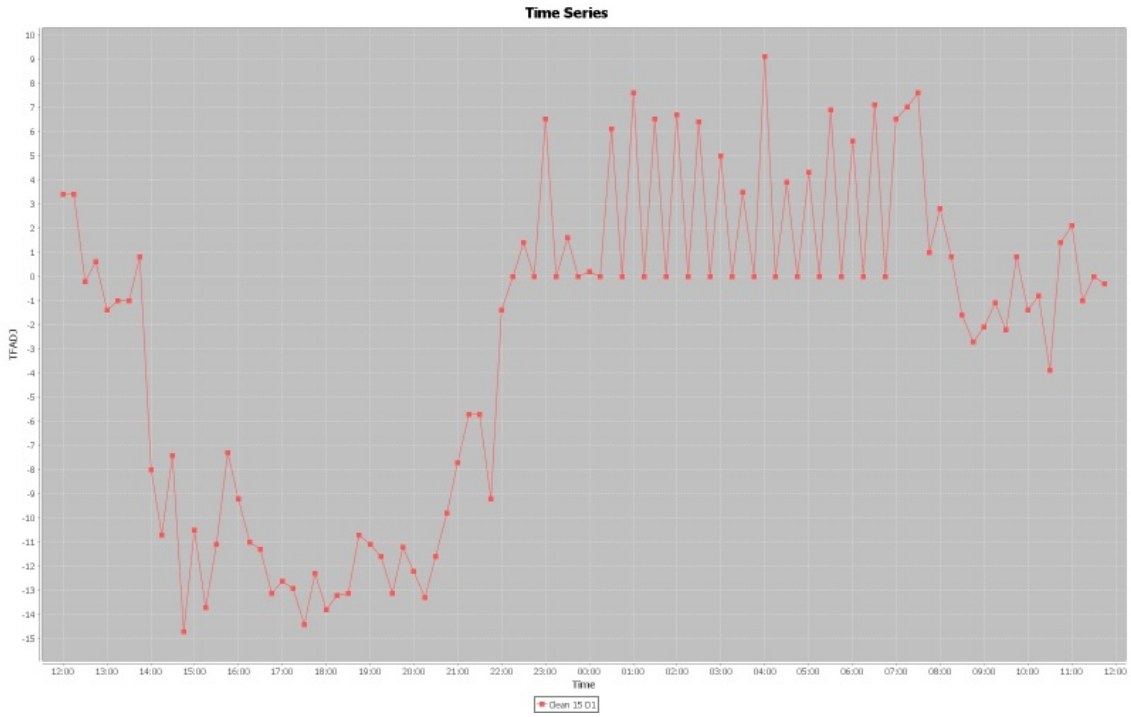
$$S = \sum_{i=1}^{n} (y_i - \hat{y_i})^2$$

http://www.cs.put.poznan.pl/jstefanowski/aed/TPDregresjawieloraka.pdf

2.2.2 Wygładzanie (“Smoothing”)

Wygładzanie to metoda starająca się zniwelować ponadprzeciętne różnice wartości między kolejnymi pomiarami. Dzięki temu podejściu można dokładniej przyjrzeć się ogólnemu zarysowi funkcji, czy jej okresowymi trendami, kosztem precyzji. Podejście to pozwala zminimalizować ewentualne szumy z badanego zbioru. Metoda ta jako argument przyjmuje szerokość okna k, w ramach którego będą uśredniane wartości. W pierwszym kroku liczy się średnią z k pierwszych wartości szeregu, następnie przesuwa okno o jeden element i znów liczy średnią z k wartości. Wyjściowy zbiór dla n-elementowego zbioru będzie miał $n - k$ wartości.

Poniżej dla porównania zaprezentowano wykres miary TFADJ w przeciągu doby 2.1, oraz jego wygładzony odpowiednik 2.2. Jak widać operacja wygładzania zredukowała skrajne wychylenia szeregu i wyklarowała paraboliczną tendencję tego zbioru.



Rysunek 2.1: Czysty sygnał TFADJ

2.2.3 Modele ARIMA

Model ARIMA (ang. Autoregressive Integrated Moving Average) koncepcyjnie składają się z trzech części - jest to autoregresja, integracja oraz średnia ruchoma.

Autoregresja jest to idea, która skupia się na wyrażeniu bieżącej wartości na podstawie$n$ poprzednich. Dla przykładu, funkcja dla $n = 2$ wygląda w ten sposób:

$$x_t = \omega_{t-2}x_{t-2} - \omega_{t-1}x_{t-1} + \omega_t$$

Średnia ruchoma ideowo jest bardzo zbliżona do autoregresji, jednak skupia się na zaburzeniach (ang. lags) w szeregu, a nie bezpośrednio na wartościach. Wzór przedstawia

Rysunek 2.2: Wygładzony sygnał TFADJ

się analogicznie jak w autoregresji.

Integracja z kolei pozwala na zastosowanie modelu ARIMA do procesów niestacjonarnych, które da się sprowadzić do procesów stacjonarnych dzięki przekształceniu oryginalnego sygnału na różnice (ang. difference equations) pomiędzy wartością obecną, a poprzednią.

Podsumowując, modele ARIMA doskonale spisują się w prognozowaniu wartości, bazując na danych historycznych szeregu, jednak nie są najlepszym wyborem jako metoda klasyfikacji. Ponadto metoda nie jest najłatwiejsza matematycznie. Biorąc pod uwagę te argumenty, metoda ta została odrzucona.

2.2.4 Analiza spektralna (widmowa)

Celem analizy spektralnej (ang. spectral analysis) szeregów czasowych jest zidentyfikowanie najczęściej powtarzających się wzorców w czasie, a następnie przybliżenie danego szeregu do procesu okresowego określonego wzorem:

$$x_t = U_1 \cos(2\Pi\omega t) + U_2 \sin(2\Pi\omega t)$$

gdzie amplituda funkcji to $A = \sqrt{U_1^2 + U_2^2}$.

Jak widać jest to kolejn metoda, za którą stoi solidne matematyczne zaplecze i która wymierzona jest w szeregi cykliczne, z którymi w zbiorze danych pacjentów jaskry nie mamy do czynienia, zatem metodę tę odrzucono.

2.3 SAX (Symbolic Aggregate approXimation)

Już na początkowym etapie rozważań nad zakresem niniejszej pracy zadecydowano, że jednym głównych nurtów jaki przyjmie, będzie przekształcanie sygnału według metody SAX. Mówiąc po krótce, jest to metoda odzwierciedlająca oryginalny sygnał w ciąg znaków, co stwarza możliwości klastfikacji danych za pomocą istniejących już algorytmów tekstowych.

2.3.1 Założenia

Korzystając z pomysłu prof. Eamonna przyjęto założenie, że dla szeregów czasowych o równych odstępach czasowych między pomiarami, można pominąć dane dotyczące momentu, w którym pomiaru dokonano i uprościć ten szereg do sekwencji samych wartości. W następujących rozdziałach przyjęto że operuje się na zbiorze N-elementowym o postaci $x_1, x_2, \ldots, x_n$

2.3.2 Schemat działania

Stworzenie reprezentacji SAX dla szeregu czasowego składa się z trzech etapów:

2.3.2.1 Z-normalizacja szeregu czasowego

Na potrzeby przedstawienia wzoru dla odchylenia standardowego przyjęto następującą definicję sumy elementów zbioru:

$$s_j = \sum_{k=1}^{N} x_k^j$$

dzięki któremu można przedstawić wzór na odchylenie standardowe tak jak poniżej[2]:

$$std = \sqrt{\frac{Ns_2 - s_1^2}{N(N-1)}}$$

Mając obliczone odchylenie standardowe i średnią wartość dla zbioru, którą wyrażono poniżej symbolem $mean$ można przejść do właściwej normalizacji, co oznacza przekształcenie każdego z elementów zbioru w sposób przedstawiony poniżej:

$$x_i = \frac{x_i - mean}{std}$$

2.3.2.2 Dyskretyzacja sygnału za pomocą algorytmu PAA (Piecewise Aggregate Approximation)

Jednym z argumentów, jaki przyjmuje algorytm SAX jest długość łańcucha wyjściowego - co w praktyce sprowadza się do tego na ile części podzielić długość szeregu czasowego. Oznacza to, że metoda zmniejsza wymiarowość szeregu z N do żądanych M elementów. Przebieg algorytmu jest intuicyjny - sekwencja $x_1, x_2, \ldots, x_N$transformowana jest do postaci $y_1 y_2, \ldots, y_M$ poprzez podział sekwencji N-elementowej na M równych części według

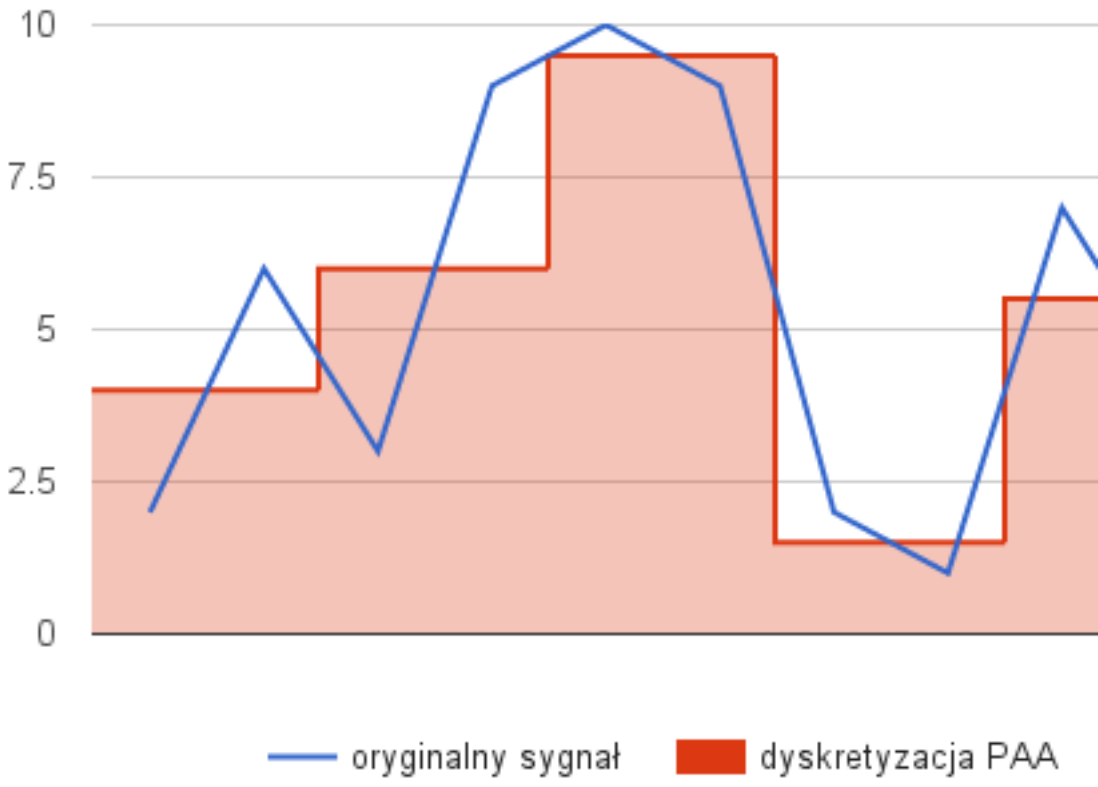
wzoru[4]:

$$y_i = \frac{M}{N} \sum_{j=\frac{N}{M}(i-1)+1}^{(N/M)i}$$

Poniżej przedstawiono przykładową dyskretyzację dla następującego szeregu czasowego:

i	1	2	3	4	5	6	7	8	9	10
oryginalny sygnał x_i	2	6	3	9	10	9	2	1	7	4

dzieląc szereg dziesięcioelementowy na pięć równych części algorytmem PAA[1]: Dzięki



Rysunek 2.3: Porównanie oryginalnego szeregu czasowego i jego 5-elementowej dyskretyzacji

temu otrzymano redukcję pięciowymiarową:

i	1	2	3	4	5
zdyskretyzowany sygnał y_i	4	6	9.5	1.5	5.5

2.3.2.3 Algorytm SAX

Jak wspomniano wyżej, jednym z argumentów jakie przyjmuje algorytm SAX jest długość sekwencji wyjściowej, zaś drugim - liczba dostępnych liter w alfabecie. Ta liczba precyzuje

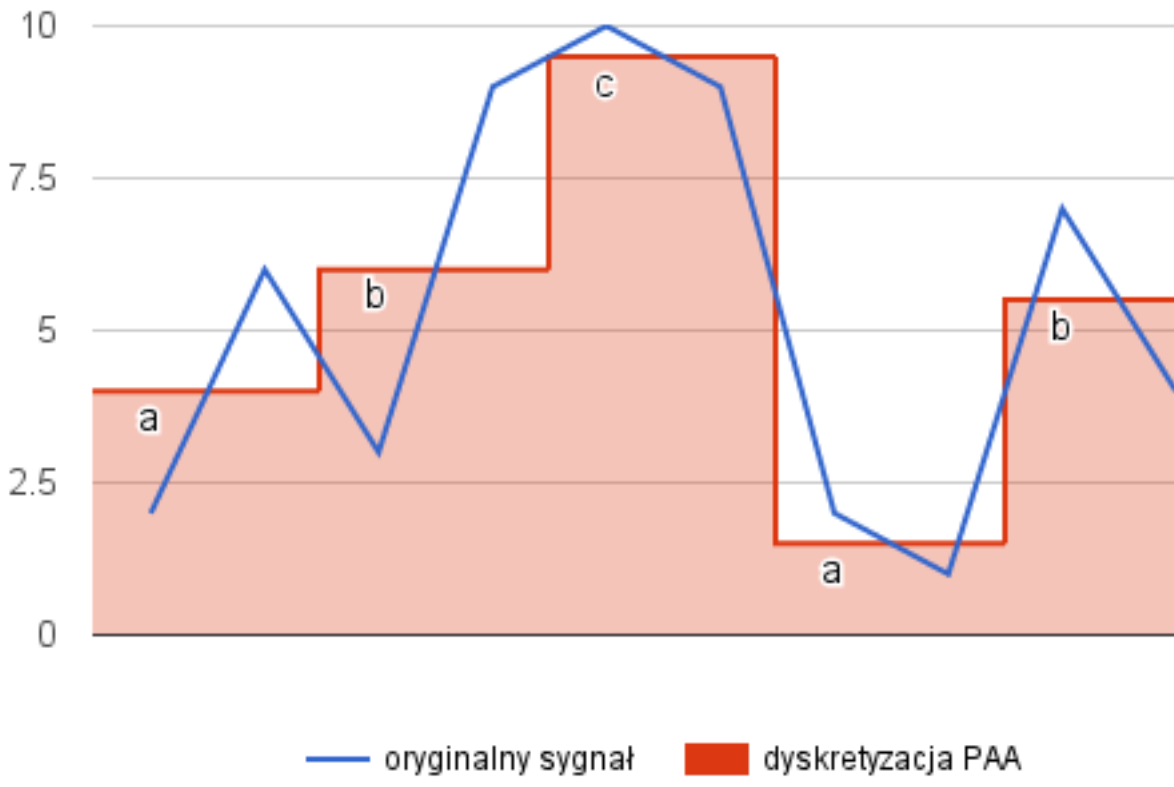
na ile części podzielić zbiór wartości danego (przetworzonego już) szeregu, a tym samym jakie indeksy (znaki) przypisać wartościom lądującym w poszczególnych zakresach. Do przypisania znaków dla odpowiednich przedziałów potrzebny jest przedział danych zatem obliczyć różnicę między elementem minimalnym a maksymalnym zbioru y

$$d = |max(y) - min(y)|$$

a następnie ustalić krok k, ustalający rozmiar okna, dzieląc zbiór wartości d przez liczbę części M. W ostatnim kroku, począwszy od wartości minimalnej przydzielamy kolejne przedziały kolejnym znakom. Dla powyższego przykładu przyjęto alfabet trójelementowy $alpha = a, b, c$. Dla $d = 9.5 - 15 = 8$ wielkość okna wynosi $k = \frac{8}{3} = 2\frac{2}{3}$. Co prowadzi do podziału:

a	b	c
$< 1\frac{1}{2}, 4\frac{1}{6})$	$< 4\frac{1}{6}, 6\frac{5}{6})$	$< 6\frac{5}{6}, 9\frac{1}{2} >$

dzięki czemu otrzymujemy przypisania naniesione na wykres: co oznacza, że ostatecz-



Rysunek 2.4: Oryginalny sygnał z dyskretyzacją i naniesionym SAXem

nym wynikiem metody SAX dla szeregu czasowego x przy zadanych parametrach długości łańcucha wyjściowego 5 oraz alfabetu trójelementowego jest ciąg znaków *abcab*.

2.4 Badane zbiory danych

Jak wspomniano wcześniej, podstawowy zbiór, który miał zostać zbadany to pacjenci z jaskrą (lub bez). Z tego powodu będzie on opisany nieco szerzej niż pozostałe zbiory

2.4.1 Pacjenci

Zbiór dostarczony przez dr. J. Błaszczyńskiego. Zbiór zawiera pomiary dla 116 pacjentów, z których 65 to pacjenci zdrowi, a 51 to pacjenci ze zdiagnozowaną jaskrą. Dla każdego pacjenta zebrano 288 pomiarów (pomiar co 5 minut przez całą dobę) tzw. TFADJ <tu rozwinięcie?> będącym przekształceniem zmierzonego w danym momencie ciśnienia w oku. Poniżej poglądowe zdjęcie metody zbierania pomiarów:

Rysunek 2.5: Metoda zbierania pomiarów od pacjentów

2.4.2 Dane prof. Eamonna Keogh'a[3]

Zbiory pobrane od prof. Eamonn'a nie zawierają opisu poszczególnych zbiorów, dlatego też ich tu nie przedstawiono. Jednak przetwarzając dane można było wyciągnąć z nich pewne interesujące statystyki, które przedstawiono w poniższej tabeli (w ostatnim wierszu porównawczo zestawiono zbiór dr. Błaszczyńskiego):

Tablica 2.1: Szczegóły danych

nazwa zbióru	**liczba instancji**	**liczba klas**	**podział na klasy**	**długość szeregu**
ECG200	200	2	'-1' - 67 rekordów '1' - 133 rekordów	96
ECGFiveDays	884	2	'1' - 442 rekordów '2' - 442 rekordów	136
TwoLeadECG	1162	2	'2' - 581 rekordów '1' - 581 rekordów	82
Yoga	3300	2	'1' - 1530 rekordów '2' - 1770 rekordów	426
MoteStrain	1272	2	'2' - 587 rekordów '1' - 685 rekordów	84
ItalyPowerDemand	1096	2	'1' - 547 rekordów '2' - 549 rekordów	24
ChlorineConcentration	4307	3	'1' - 1000 rekordów '3' - 2307 rekordów '2' - 1000 rekordów	166
Two Patterns	5000	3	'2' - 1248 rekordów '3' - 1245 rekordów '4' - 1201 rekordów '1' - 1306 rekordów	128
Wafer	7174	2	'1' - 6402 rekordów '-1' - 762 rekordów	152
InlineSkate	650	7	'2' - 100 rekordów '3' - 103 rekordów '7' - 62 rekordów '6' - 98 rekordów '4' - 108 rekordów '5' - 117 rekordów '1' - 62 rekordów	1882
Pacjenci	*116*	*2*	*'0' - 65 rekordów* *'1' - 51 rekordów*	*288*

2.5 Special characters

1. Non-breaking space can be inserted using **Ctrl-space**. It produces "`~`" in LaTeX code.
2. A normal, inter-word space can be inserted using **Ctrl-Alt-space**. It produces "`\ `" in LaTeX code. This type of space is useful for formatting spacing after dots, e.g. here. By default LaTeX produces here a longer space used for separating whole sentences.
3. A thin space can be produced by **Ctrl-Shift-space**, e.g. here. It produces "`\,`" in LaTeX code.
4. Sentence-ending space can be inserted using **Ctrl-.**, which produces "`\@.`" in LaTeX code. This type of space is useful in sentences ending with a capital letter. In such cases LaTeX recognizes the last word as a acronym and places a regular inter-word space instead of inter-sentence space. Consider the following example:

   This can be achieved by using HTTP. This protocol...

5. Hyphenation indicator can be inserted using **Ctrl- –**, which is used for marking possible places of hyphenation, e.g. democracy.

2.6 Source code examples

There are a few different methods of including sample codes:

1. Using standard LyX-Code style:

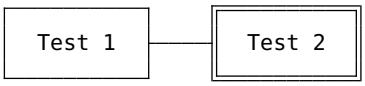
```
#include <stdio.h>

int main() {
  printf("Hello world!\n");
  return 0;
}
```

   Note 1: Empty lines must contain at least one single space to remain visible.

   Note 2: There is no way to activate automatic syntax highlighting inside LyX-Code. However, you can use normal inline formatting inside.

   Note 3: Lyx-Code can contain special characters, so it can be used to produce some ASCII art, e.g.:



2. By inserting *Program Listing*:

```
#include <stdio.h>

int main() {
  printf("Hello world!\n");
  return 0;
}
```

   Note: By default the lstlisting environment does not add any left margin. You can change it by adding xleftmargin in the *Settings* ▷ *Advanced* dialog box, e.g.:

```
    procedure sayHello()
```

3. By inserting LaTeX Code (ERT block) and using codeblock environment:

```
#include <stdio.h>

int main() {
  printf("Hello world!\n");
  return 0;
}
```

4. The listings package can produce floats by itself. See Listing. 2.1 for example.
5. And finally, You can include code from exte
6. rn
7. al file:

```
\documentclass[11pt,a4paper,polish,thesis]{dcsbook}
```

Listing 2.1: The Hello World program in C

```c
#include <stdio.h>

int main() {
  printf("Hello world!\n");
  return 0;
}
```

```
\usepackage[utf8]{inputenc}
\usepackage{babel}
\setcounter{secnumdepth}{4}
\setcounter{tocdepth}{3}

\begin{document}
```

Rozdział 3

Concept and Design of the System

Rozdział 4

Implementation

4.1 Koncepcja budowy biblioteki

Wiodącym założeniem przyświecającym pracom implementacyjnym nad biblioteką obliczeniową było podążanie za dobrymi praktykami wytwarzania oprogramowania według Roberta C. Martina - z głównym naciskiem na OCP (Open Closed Principle)[5] - zasadę „otwarte-zamknięte". Oznacza to, że oprogramowanie ma być otwarte na zmiany, lecz zamknięte na modyfikacje - starano się zatem osiągnąć stan, w którym istniejący kod można rozszerzyć z jak najmniejszym wysiłkiem wniesionym w edycję istniejącego kodu. Szczegóły tej idei przedstawiono w poniższych sekcjach.

4.1.1 Użyte biblioteki pomocnicze

Żeby móc w pełni opisać funkcjonalność i sposób budowy programu należy nadmienić jakimi bibliotekami wspomagano się podczas tego procesu.

4.1.1.1 Weka

Najważniejsza biblioteka, z którą program współpracuje najściślej. Wiedząc, że Weka jest często używana do klasyfikacji czy uczenia maszynowego, autor chciał stworzyć narzędzie kompatybilne z programem stworzonym na Uniwesytecie w australijskim Waikato. Kilka kluczowych klas programu autora rozszerza klasy pakietu `weka` stwarzając możliwości wykorzystania rozmaitych składowych tego systemu. Ponadto używanie klas takich jak `weka.classifiers.Evaluation` do przeprowadzania eksperymentów daje pewność, że wykorzystywane rozwiązanie jest dobrze przetestowane i powinno zwracać rzetelne wyniki.

4.1.1.2 jMotif

Z pakietu `jMotif` zaczerpnięto metody zajmujące się tworzeniem łańcucha danych na podstawie zadanego szeregu czasowego za pomocą algorytmu SAX opisanego w rozdziale **??**.

4.1.1.3 JFreeChart

Biblioteka służąca do rysowania wykresów. Bardzo pomocna w początkowej fazie implementacji, kiedy autorowi zależało na zobrazowaniu przebiegu szeregów szasowych. Ich

wizualizacja mogła przyczynić się do lepszego zrozumienia zależności między przebiegami, a diagnozą pacjenta.

4.1.1.4 Jxl

Dzięki pakietowi `jxl`w początkowej fazie implementacji autor mógł ściśle współpracować z arkuszami kalkulacyjnymi w celu raportowania danych do plików .xls w celu ich dalszej analizy statystycznej.

4.1.1.5 joda-time

W momencie gdy prace nad tą pracą dyplomową zostały rozpoczęte, wspracie dla obiektów typu data czy czas było dla autora niewystarczające, dlatego zdecydowano się na wykorzystanie biblioteki `joda-time` w celu zapewnienia wymaganej obsługi tego rodzaju danych.

4.1.1.6 fast-dtw

Pakiet fast-dtw dostarcza obsługę obliczeniową dla eksperymentu bazującego na algorytmie Dynamic Time Warping, dzięki czemu autor mógł skorzystać z gotowego, przetestowanego i relatywnie wydajnego czasowo i pamięciowo rozwiązania dla tego celu.

4.1.1.7 apache-commons

W momentach gdy brakowało pewnych struktur danych w obecnej wersji Java - takich jak choćby Pair (ang. para) lub należało skorzystać z popularnych metod takich jak obliczenie regresji, korzystano z bibliotek apache-commons.

4.1.2 Punkt wyjścia

Praca programu zorientowana jest wokół obiektu `Workflow` (ang. „przepływ”, „proces”) - ma on na celu zdefiniowanie następujących po sobie kroków składających się na przebieg eksperymentu - począwszy od wczytania danych, aż do wypisania ostatecznego wyniku. Jego rdzeń zawiera się w abstrakcyjnej klasie `WorkflowBase`która posiada trzy główne metody - odpalenia przebiegu pełnego eksperymentu, wygenerowania pliku arff dla eksperymentu oraz wyliczenia wyniku eksperymentu dla zadanego pliku arff. Ponadto jest w stanie opisać podstawowe statystyki wczytanych danych, takie jak liczba klas decyzyjnych, podział instacji w klasach, czy liczba atrybutów w ramach instancji. Aby stworzyć własny eksperyment wystarczy stworzyć klasę dziedziczącą z `WorkflowBase` oraz zaimplementować wymagane metody. Przykładowy szkic takiej klasy przedstawiono na listingu 4.1: Jeżeli wczytane przez nas dane wymagają jakiejkolwiek ingerencji obliczeniowej - tak jak na przykład eksperyment Counted, to fragment kodu odpowiadający za te obliczenia powinien znaleźć się w metodzie `processData`. Jak widać w przywołanym listingu, wszystkie wymagane kalkulacje odbywają się w tym fragmencie kodu. Kolejna metoda to `buildInstances` która odpowiada za ustawienie odpowiedniego eksportera danych (w szczegółach zostanie omówiony niżej) oraz przekazanie mu przeliczonych danych. Za jego pomocą tworzony jest obiekt `Instaces`, który musi zostać zwrócony jako wynik metody. Funkcja

Listing 4.1: Kod eksperymentu Counted

```java
package pl.poznan.put.TimeSeries.Workflows;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

import org.apache.commons.lang3.tuple.Pair;

import pl.poznan.put.TimeSeries.Constants.DivisionOptions;
import pl.poznan.put.TimeSeries.DataExporters.CountedSaxArffBuilder;
import pl.poznan.put.TimeSeries.DataProcessors.PeriodicNgramCounter;
import pl.poznan.put.TimeSeries.Model.CalculatedRecord;
import pl.poznan.put.TimeSeries.Model.IRecord;
import pl.poznan.put.TimeSeries.Util.Config;
import pl.poznan.put.TimeSeries.Util.DataDivider;
import weka.core.Instances;

public class CountedWorkflow extends WorkflowBase {

    private List<CalculatedRecord> calculatedRecords;

    public CountedWorkflow(DivisionOptions divisionOption, boolean glaucoma) {
        super(divisionOption, glaucoma);
    }

    @Override
    protected Instances buildInstances() {
        exporter = new CountedSaxArffBuilder(calculatedRecords);
        return exporter.buildInstances();
    }

    @Override
    protected void processData() throws Exception {
        calculatedRecords = new ArrayList<CalculatedRecord>();

        for (IRecord record : records) {
            ArrayList<HashMap<String, Integer>> periodicallyCountedNgrams =
                                    new ArrayList<HashMap<String, Integer>>();

            List<String> dividedSax = DataDivider.divideStringRegularly(
                    record.getSaxString(), divisionPartsAmount);

            for (String elem : dividedSax) {
                HashMap<String, Integer> ngramCountMap = PeriodicNgramCounter
                        .slashStringAndCountNgrams(elem, windowLen);
                periodicallyCountedNgrams.add(ngramCountMap);
            }

            CalculatedRecord calcRecord = new CalculatedRecord(
                    periodicallyCountedNgrams, record.getDestinationClass());
            calculatedRecords.add(calcRecord);
        }
    }

    @Override
    protected void setConcerningParams() {
        concerningParameters.add(Pair.of("parts", divisionPartsAmount));
        concerningParameters.add(Pair.of("ngram", windowLen));
        concerningParameters.add(Pair.of("alpha", Config.getInstance().getSaxAlphabeatSize()));
    }
}
```

`setConcerningParams`służy już tylko do tego, aby podczas wyświetlania wyniku eksperymentu wypisać, które parametry programu mają wpływ na pracę tego eksperymentu, oraz jakie ich wartości zostały ustawione podczas bierzącego wykonania.

4.1.3 Importowanie danych

Importowanie danych zostało rozdzielone ze względu na dwa różne formaty danych jakie potrafi wczytać program. Ostateczną wersją danych pacjentów jest opisanie danych każdego pacjenta w osobnym pliku `.csv`. W każdym z nich podany jest identyfikator pacjenta, oraz dane z poszczególnych pomiarów ciśnienia w oku. W ramach każdego pomiaru udostępnionych jest sporo danych, takich jak godzina o której pomiar został zdjęty, pozycja ciała w której znajdował się pacjent itp. choć na potrzebę tej pracy korzystano tylko z miary `TFADJ`. Jedyna informacja, której nie ma w tym pliku to diagnoza - ta opisana jest w pliku zbiorczym dla wszystkich pacjentów. W kolejnych wierszach znajdują się pary identyfikator pacjenta - diagnoza.

Drugim formatem, jaki wspiera program jest format danych prof. Eamonna. Ich charakterystyka jest prosta - każdy nowy wiersz w pliku to nowa instacja. W ramach każdego wiersza figurują atrybuty oddzielone spacjami, z których pierwszy to atrybut decyzyjny, a kolejne to szereg wartości składające się w szereg czasowy dla tej instancji. Tak jak wspomniano wcześniej, zapis ten pomija dokładny czas zebrania pomiaru. Klasy odpowiadające za wczytywanie danych to odpowiednio `PatientDataImporter2` oraz `DataImporterEamonn`.

4.1.4 Eksportowanie pliku arff

Rozdział 5

Performance Evaluation

Rozdział 6

Conclusions

Dodatek A

Users Guide

Bibliografia

[1] Opis przeprowadzenia algorytmu paa w bibliotece jmotif. http://jmotif.github.io/sax-vsm_site/morea/algorithm/PAA.html. Ostatni dostęp: 2015-09-26.

[2] Wzór na odchylenie standardowe. https://en.wikipedia.org/wiki/Standard_deviation#Rapid_calculation_methods. Ostatni dostęp: 2015-09-26.

[3] Yanping Chen, Eamonn Keogh, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, and Gustavo Batista. The ucr time series classification archive, July 2015. www.cs.ucr.edu/~eamonn/time_series_data/, ostatni dostęp 24.09.2015.

[4] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. A symbolic representation of time series, with implications for streaming algorithms. pages 2–11, 2003.

[5] Robert C. Martin. Design principles and design patterns. = http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.94.8854.

[6] Robert H. Shumway and David S. Stoffer. *Time Series Analysis and Its Applications: With R Examples (Springer Texts in Statistics)*. Springer, 2010.